Network Failure Analysis for CENIC

***Abstract*—**

I. Introduction

As more and more enterprises heavily rely on the networks, there is clearly an increase in importance to provide better availability or even uninterrupted service more aggressively. However, it brings us a much challenging work to deliver such promise as we are building much more complex and larger networks comparing to years ago. Although tons of research projects have been trying to work it out, it is still not very well solved. As well as many previous work shown [2]–[4], we believe the better analysis of the network failures should be considerably helpful for us to understand how the failures occur, how they affect the network behaviors and how to eliminate them in order to provide better availability.

To deliver such network failure analysis in practice, one will need much information about the causes, time lasted, influence to the network and many other specifics that are clearly not intuitively provided by the network protocols in use currently. Thus, the traditional approaches [1], [5] for doing this kind of analysis is to build some extra software or hardware support, which would incur large amount of expense and perhaps performance overhead. Consequently, most of such failure analysis are performed in the research community. To our knowledge, California Fault Lines [6] is the first piece of work that trying to extract these information from commonly used production networks today.

Other than reconstructing historical network failure events, we are able to perform further characterization and analysis on those failures with very similar "cheap and dirty" data which we can easily obtain from today's networks. To better understand the network behavior, we describe and validate a methodology to map the routing changes to corresponding ISIS failures. In addition, we propose an approach to better understand how the failures occur and how they affect the network behaviors. Furthermore, characterization and discussion are presented about the unexpected network behaviors including routing loops and non-existent links.

Specifically, we set up six source machines respectively at UC Berkeley, UC San Diego, UC Los Angeles, UC Santa Barbara, UC Davis and UC Santa Cruz, all in CENIC (the Corporation for Education Network Initiatives in California) which is the autonomous system we are trying to analyze. Each of the six machines are supposed to send a series of traceroute requests to the end hosts of the failing link when it is detected with the help of syslog messages automatically sent by CENIC. Firstly, we map the traceroute data back to corresponding network failures for further study, and validate how well the failure detection and traceroute are working based on the mapping. Secondly, we detect the routing loops and non-existent links in the traceroute data, and carry out some statistics and characterization on these unexpected network behaviors as well as the ISIS failures. At last, with the help of link weights of CENIC, we evaluate how well the network protocols work on routing and analyze how the failures affect the network behaviors.

The rest of the paper is organized as follows. We begin with the discussion of related work in Section II. Then the data we use in this work is introduced in Section III. Section IV presents the our methodology for the characterization and analysis, and they are validated in Section V. We present our analysis in Section VI and conclude in Section VII.

II. Related work

bla bla

III. Data sources

bla bla

IV. Methodology

A.

how to reconstruct the network failure history

B.

how to map the failure with traceroute

C.

loop and non-existent link detection

V. Validation

A. Network History

Assuming the active probing of the certain network could give us a complete history of it, we had this history of the CENIC from 10/20/2012 to 02/09/2013. There were 40074 ISIS failures during this period in total.

Since we would like to do a comparison among the route before, during and after the failure to better understand how failure affects the network behaviors, it was useful to find out how long each failure last for. Thus, the CDF of failure recovery time was shown in Figure 2. Surprisingly, there were 77.87% of the failures that were recovered in 0 seconds, which means they didn't actually affect the network in the granularity of second that we used. And more than 95% failures were recovered within 6 seconds, which could potentially make the corresponding traceroutes much less interesting.

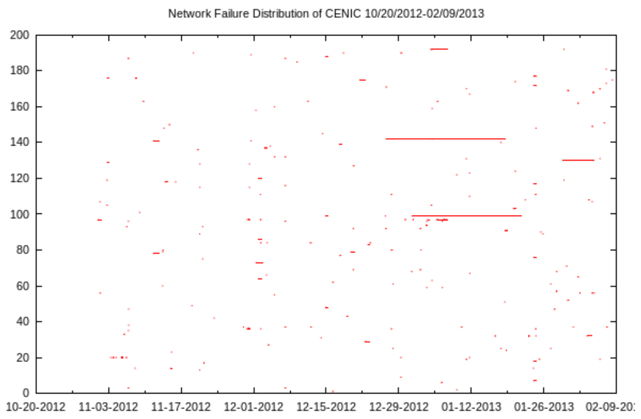

Fig. 1. Network failures distribution of CENIC from 10/20/2012 to 02/09/2013, where Y-axis is the failure link. Most of the failures lasted for a fairly short time period and there were 192 links had failed at least once during this time period.

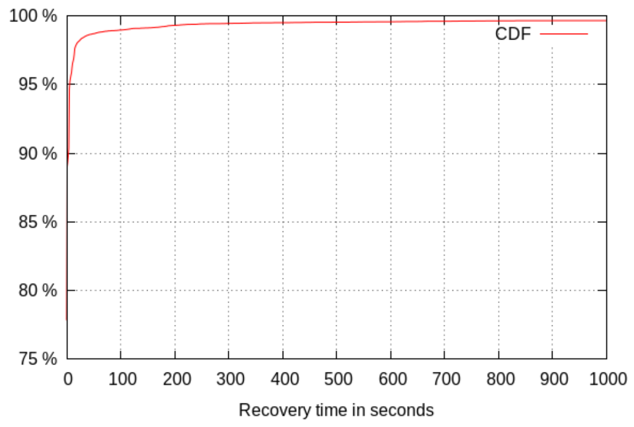

Fig. 2. Network failure recovery time CDF of CENIC where more than 95% failures recovered within 6 seconds.

Because of the relatively short recovery time, it is very likely that the failure had already recovered before we sent very few traceroutes. Thus it could reduce the amount of data we were interested in because most of the traceroute would not even aware of the route changes due to the large granularity. However, even with small percentages, we didn't get very tiny numbers in the following analysis because of the fairly large number of failures we had in total.

B. Network History Reconstruction

To validate the network history we reconstructed from the traceroute data, we tried to map the probed failures to corresponding traceroutes. Based on the link and router map of CENIC we had successfully mapped 40073 ISIS failures to corresponding traceroute data out of the total 40074. The only one that had not been mapped was because we didn't have the IP address of the link. Thus, we believed it is save to say our failure detection works pretty well from this point.

For these 40073 detected failures, we had pinged at least one end of the failure link. And for 39505 out of 40073 (98.58%) failures, we had pinged both ends at least once. Consequently, we believed our data is very representative for our analysis.

C. Statistices of Unexpected Network Behavior (Loops & Non-existent Links)

In the 70831 traceroute records, we found 366 records that contain loops in total. We categorized these 366 records by the length of loop and found that more than 95% (350 out of 366) of loops are jumping between only two routers. As of the long loop records, most of them would also fall into a two-router trap finally.

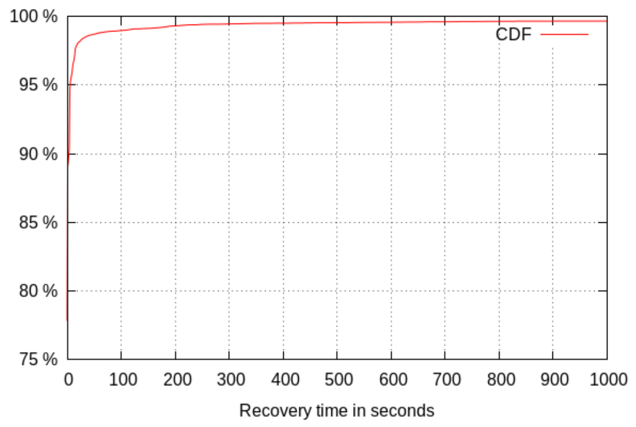

Fig. 3. Loop Length Statistics

D. Measurement Bias

First of all, the granularity of the network failures and the traceroute data could potentially affect our analysis of the network behaviors. As discussed ealier, the ISIS failures were probed at the granularity of one second, while many of them lasted less than couple seconds actually. This reduced the number of interesting traceroutes since they could not even reflect the effects of such short failures. We believed that a much finer granularity of the failures would be very helpful for further analysis.

Additionally, our failure detection mechanism is based on adjacency status reported by the underlying routing protocol. For example, to ensure connectivity, the IS-IS protocol requires routers to send and receive hello messages. By default, a router sends a hello message once every ten seconds and declares a link disconnected if no hello message is received for thirty seconds. Consequently, the failure detection could have up to 30 seconds of delay, which could reduce the accuracy of the mapping between the failure and corresponding traceroutes. In addition, this would probably affect our analysis about how the failures affected the network behaviors.

VI. Data Analysis

A. Loop Characterization

Given the traceroute data, we can figure out

B. Non-Existent Links Charaterization

Among the 70831 trace-route records, there were 80 records has non-existent links in them. We classify these records into 5 categories, with respect to the complexity of their cause. They are: normal route shift, complex route shift, shortcut, trapped in loop, cannot verify.

Normal Route Shift

Consider a trace-route with max hops 5. In most situation, a certain route won't change during the process of trace-route. However, it is possible for router to choose another path during this process. An example is shown in figure: 4.

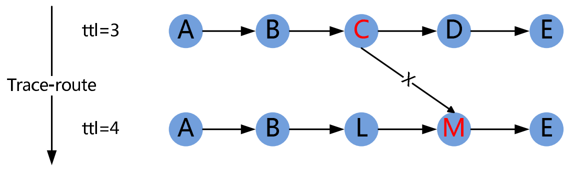

Fig. 4. A Normal Route Shift example

It is important to notice in figure:4 that, both of the two routes reached destination E, with C on the third hop and M on the fourth hop. We identify this kind of links by finding valid route with the same node on the same hop. 45 out of the 80 records fall into this category.

Complex Route Shift

This was a more complicated situation than in the first case. There were three key differences with the first category. First, a complex route shift may shift among more than 2 routes in a trace-route. Second, the picked route is allowed to be a route that did not reach the destination. Third, it is possible that none of other trace-route record can match a node on a certain hop of a complex route shift. Still, consider a trace-route with max hops 5. An example is shown in figure: 5.

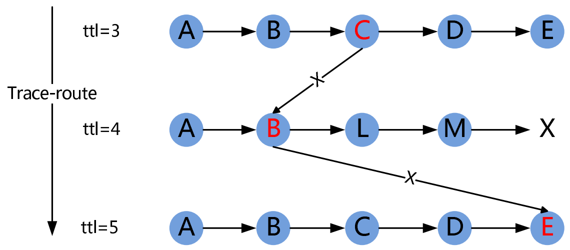

Fig. 5. A Complex Route Shift example

From figure:5, we cannot tell what exactly happened. Although we see node B on hop 4, can see no B on hop 4 from any other record. This means, we have no idea of how and why trace-route reached B on the fourth hop. Hop 5 behaves similar with normal route shift. There were 9 records fall into this category.

Shortcut

Short cut is a special case of a normal route shift, with one of the route in the shift to be an invalid route. An example is shown in figure:6.

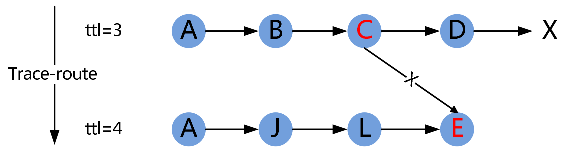

Fig. 6. A Shortcut example

To explain this phenomenon in a more intuitive way, suppose one of the router is down, trace-route cannot get to the destination due to the router failure. Then, during the trace-route, the router is fixed and back to work, trace-route got the destination directly on the next hop. There were 3 records fall into this category.

Trapped in loop

This is another special case of normal route shift, with one of the route is a loopy route. An example is shown in figure:7.

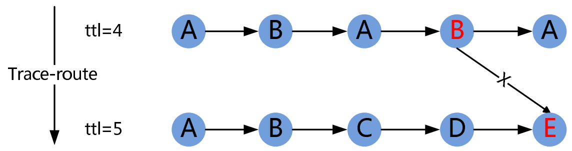

Fig. 7. Route Trapped in Loop example

There were 15 records fall into this category. **Cannot verify** The rest 8 records fall into this category. For these records, there exist some node that is never seen in other trace-route records with the same source and destination. Thus, there is no way we can tell what happened with such records.

C. Failure Analysis

Given the traceroute data, we were able to take a closer look at the ISIS failures, which could potentially help us better understand the network behaviors and then try to reduce the number of failures. From all the 40073 failures we've ever pinged, only 125 (3.11%) of them had ever caused a route change from at least one source. This was a relatively small number comparing with the total failures we detected, which I thought is mainly because we only had 6 sources to send traceroute requests and there are only 3 of them worked as we expected, as well as most of the failures lasted for very short time period. It would be considerably helpful if we could have more sources sending traceroute to the failure end hosts, which would probably give us more interesting routing changes.

And about half of these changes, 63 out of 125 (50.40%), were not detectable because we could only get more or less unrecognized hops displayed as ** from traceroute. Since we couldn't recognize them, we didn't have their route weights and were not able to draw any conclusions on how their routes

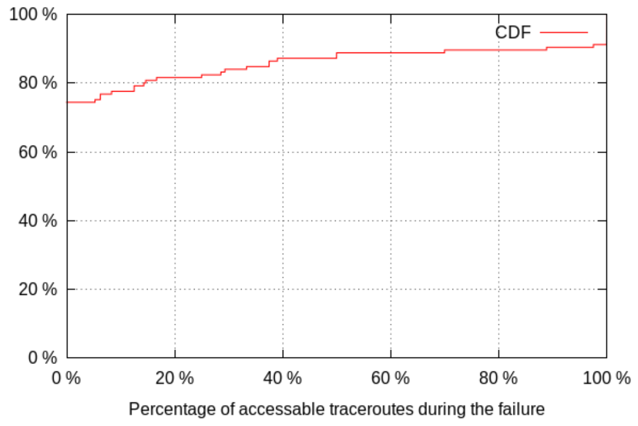

Fig. 8. Percentage CDF of accessible pings during the link failure time. 74.40% of the 125 failures which had caused route changes hadn't affected the accessibility of the end hosts, which meant the routing protocaol worked pretty well during those failures.

change quantitatively. But we could still do some analysis about the accessibility of the failure ends.

It was clear in Figure 8, that the end hosts of 93 out of 125 (74.40%) failures have kept reachable during the failure period. Because all these 125 failures had caused route changes, which is to say that all these routes were affected by the failures somehow, it was relatively well that the routing protocols were performing. However, there were still 8.80% of the failures caused some of the end hosts unreachable during the whole failure period.

Furthermore, we did some statistics for the failure links, where a few links (192) occurred large number of failures (40073). From Figure 9, we could easily find that most of the failure links had small numbers of failures ever occurred. But the most surprising part of this figure is that there are some links that had more than thousands of failures during this period of time, which were supposed to be taken care of after tens of frequent occurrences.

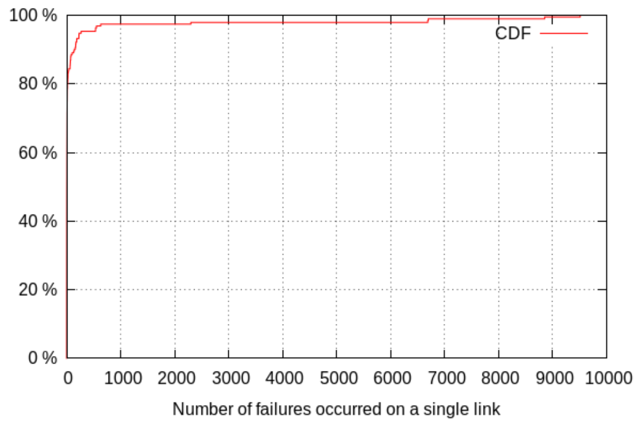

Fig. 9. The distribution CDF of the failure links. Most of the failure links had very small numbers of failures, where 74.48% of them had occurred less or equal 10 failures. However, there are some links had very frequent failures that were more than 8000 times during this period.

D.

routing performance characterization

E.

* how failure affects routing (better be extended into several sections)

VII. Conclusion

Acknowledgment

We thank Professor Scott Baden for his awesome lecture notes for CSE 260, and the usage of Bang HPC Cluster where we did our experiments. Also, we thank the University of Tennessee of Knoxville for the Basic Linear Algebra Subprograms.

References

[1] Ítalo Cunha, Renata Teixeira, Darryl Veitch, and Christophe Diot. Predicting and tracking internet path changes. *SIGCOMM-Computer Communication Review*, 41(4):122, 2011.

[2] Craig Labovitz, Abha Ahuja, and Farnam Jahanian. Experimental study of internet stability and backbone failures. In *Fault-Tolerant Computing, 1999. Digest of Papers. Twenty-Ninth Annual International Symposium on*, pages 278–285. IEEE, 1999.

[3] Athina Markopoulou, Gianluca Iannaccone, Supratik Bhattacharyya, Chen-Nee Chuah, Yashar Ganjali, and Christophe Diot. Characterization of failures in an operational ip backbone network. *IEEE/ACM Transactions on Networking (TON)*, 16(4):749–762, 2008.

[4] Venkata N Padmanabhan, Sriram Ramabhadran, Sharad Agarwal, and Jitendra Padhye. A study of end-to-end web access failures. In *Proceedings of the 2006 ACM CoNEXT conference*, page 15. ACM, 2006.

[5] Vern Paxson. End-to-end routing behavior in the internet. *Networking, IEEE/ACM Transactions on*, 5(5):601–615, 1997.

[6] Daniel Turner, Kirill Levchenko, Alex C Snoeren, and Stefan Savage. California fault lines: understanding the causes and impact of network failures. *ACM SIGCOMM Computer Communication Review*, 40(4):315–326, 2010.